Journal of Air Pollution and Health (Winter 2025); 10(1): 37-60

Original Article

Available online at http://japh.tums.ac.ir

Prediction of particulate matter $PM_{2.5}$ level in the air of Islamabad, Pakistan by using machine learning and deep learning approaches

Muhammad Waqas[1,*], Shahid Noor Jan[1], Basir Ullah[1], Afed Ullah Khan[1], Ateeq Ur Rauf[1], Bakht Niaz Khan[2]

[1] Department of Civil Engineering, Bannu Campus, University of Engineering and Technology Peshawar, Bannu, Pakistan
[2] Water and Sanitation Services Bannu (WSSB), Khyber Pakhtunkhwa, Pakistan

ARTICLE INFORMATION

Article Chronology:
Received 04 November 2024
Revised 04 January 2025
Accepted 25 February 2025
Published 29 March 2025

Keywords:
Long and short-term memory (LSTM); Deep learning; Air quality; Machine learning; Multi-layers neural network (MLNN)

CORRESPONDING AUTHOR:

engr.muhammadwaqas88774@gmail.com
Tel : (+92 91) 9216796-8
Fax : (+92 91) 9216663

ABSTRACT

Introduction: Air pollution is a significant global health challenge, contributing to the deaths of millions of people annually. Among these pollutants, Particulate Matter ($PM_{2.5}$) is the most harmful to the respiratory system causing serious health problems. This study focused on predicting $PM_{2.5}$ in the air of Islamabad, capital of Pakistan by using machine learning and deep learning models.

Materials and methods: Two machine learning models (Decision Tree and Random Forest) and four deep learning models including Multi-Layer Neural Network (MLNN), Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU) are used in the study. Each model's performance was assessed by using statistical indicators including coefficient of determination (R^2), Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Relative Root Mean Square Error (RRMSE). These models are also ranked based on their performance by compromise programming technique.

Results: Machine learning models performed better in the training phase by achieving higher R^2 values of 0.98 and 0.97 but couldn't maintain the same performance in the testing phase. Whereas the deep learning models performed best in both the training and testing phases. MLNN model attained higher R^2 value of 0.98 in training and 0.88 in testing and is evaluated as top-ranked prediction model in predicting particulate matter $PM_{2.5}$. Whereas, LSTM, GRU, RNN, Decision Tree, and Random Forest are placed at the 2nd, 3rd, 4th, 5th, and 6th positions having R^2 values of 0.86, 0.87, 0.82, 0.99, and 0.97 during training and 0.71, 0.69, 0.69, 0.75, and 0.85 respectively during testing.

Conclusion: Deep learning models, especially MLNN, showed strong performance in predicting $PM_{2.5}$ as compared to the machine learning models.

Please cite this article as: Waqas M, Jan ShN, Ullah B, Khan AU, Rauf AU, Niaz Khan B. Prediction of particulate matter $PM_{2.5}$ level in the air of Islamabad, Pakistan by using machine learning and deep learning approaches. Journal of Air Pollution and Health. 2025;10(1): 37-60.

Introduction

Air quality has become a pressing global concern due to its significant impact on human health, ecosystems, and economies [1]. The degradation of air quality, particularly in urbanized and industrialized regions, has reached alarming levels. Multiple factors, including rapid population growth, increased combustion of fossil fuels, vehicular emissions, and the expansion of industrial activities, drive this decline [2]. Among the various air pollutants, particulate matter with an aerodynamic diameter of less than 2.5 μm Particulate Matter ($PM_{2.5}$) has emerged as the most hazardous to human health. Due to its microscopic size, $PM_{2.5}$ can penetrate deep into the respiratory tract, enter the bloodstream, and exert systemic effects on vital organs. Exposure to elevated concentrations of $PM_{2.5}$ has been strongly associated with respiratory illnesses such as asthma and Chronic Obstructive Pulmonary Disease (COPD), cardiovascular disorders like hypertension and heart attacks, reduced life expectancy, and increased morbidity and mortality rates [3]. According to the World Health Organization (2021), air pollution contributes to over seven million premature deaths annually, underscoring the urgent need for effective air quality management strategies [4]. Beyond its adverse health effects, $PM_{2.5}$ also contributes to reduced visibility, environmental degradation, and climate change, further emphasizing the necessity of robust monitoring and forecasting systems.

In recent years, substantial efforts have been devoted to understanding the dynamics of air pollutants and improving prediction systems. Traditional air quality forecasting methods, such as regression models, time-series analysis, and Chemical Transport Models (CTMs), have been widely utilized to predict pollutant concentrations and evaluate emission reduction policies [5]. However, these approaches often exhibit limitations, particularly when addressing nonlinear, high-dimensional interactions among meteorological factors, emission sources, and atmospheric processes [6]. Additionally, CTMs are computationally expensive and require extensive domain expertise, which limits their scalability for real-time and region-specific applications [7].

The emergence of Machine Learning (ML) and Deep Learning (DL) techniques has transformed the landscape of air quality prediction. ML models such as Random Forest (RF), Gradient Boosting Machines (GBM), and Support Vector Machines (SVM) have demonstrated strong predictive capabilities, particularly in capturing spatial and temporal variations in air quality data [8]. Meanwhile, DL models like Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) have proven effective in handling sequential and spatial data, respectively. For instance, LSTM networks excel in modeling temporal dependencies inherent in time-series data, whereas CNNs are adept at extracting spatial patterns from satellite imagery or geographic datasets [9]. Hybrid models combining ML and DL approaches have also shown promise in improving forecasting accuracy by leveraging the strengths of both paradigms [10].

Several studies have highlighted the effectiveness of ML and DL models in air quality prediction. For example, [9] applied GCN and E-LSTM to predict $PM_{2.5}$ concentrations, demonstrating their robustness in capturing nonlinear relationships between predictor variables. Similarly, [11] explored the application of LSTM models for time-series $PM_{2.5}$ forecasting, achieving superior performance compared to traditional statistical methods. Despite these advancements, critical gaps remain in existing literature. Most studies

focus on regions with extensive air quality monitoring networks, such as North America, Europe, and China, while neglecting Low- and Middle-Income Countries (LMICs), where air quality data are sparse or inconsistent [12, 13]. Additionally, limited research has conducted systematic comparative analyses of ML and DL models under varying data conditions, particularly in resource-constrained environments like Pakistan.

Islamabad, the capital of Pakistan, presents a unique case for air quality research. While often perceived as having relatively better air quality than other major cities, the city is increasingly impacted by $PM_{2.5}$ pollution due to rapid urbanization, vehicular emissions, and its proximity to industrial zones. Yet, the scarcity of high-quality, long-term air quality data poses significant challenges for effective monitoring and forecasting. Addressing this gap requires innovative solutions, including the integration of ML and DL models, the application of data augmentation techniques, and the use of transfer learning to leverage data from better-monitored regions [6].

Moreover, while studies have begun exploring advanced techniques like explainable AI (XAI), attention mechanisms, and hybrid ML-DL architectures, their application to $PM_{2.5}$ prediction remains limited [14]. These methods hold great promises for improving model interpretability, addressing data scarcity, and enhancing predictive accuracy, especially in underrepresented regions. Finally, many studies adopting hybrid ML-DL approaches fail to provide a clear rationale for their integration, limiting the practical relevance and generalizability of their findings [10].

This study seeks to address these gaps by conducting a comprehensive comparative analysis of ML and DL models for $PM_{2.5}$ prediction in Islamabad, Pakistan. The research evaluates the performance of ML models such as RF and GBM alongside DL architectures like LSTM and CNN, focusing on their effectiveness under varying data conditions. Additionally, the study explores innovative techniques like transfer learning and data augmentation to overcome data limitations and investigates the potential of hybrid ML-DL approaches for improving prediction accuracy. By addressing these challenges, this study contributes to the development of robust, accurate, and scalable air quality forecasting systems tailored to the unique needs of LMICs.

The novelty of this research lies in its dual focus: first, advancing the application of ML and DL models for air quality prediction in data-scarce regions, and second, exploring innovative solutions to overcome challenges posed by limited data availability. The findings are expected to provide valuable insights for policymakers, urban planners, and public health officials, supporting the development of evidence-based strategies for air pollution mitigation in Islamabad and similar urban settings.

Study area

Islamabad is one of the major cities and capital of Pakistan that is located between the latitude of 33.6844° North and a longitude of 73.0479° East as shown in Fig. 1. It is in the northern part of Pakistan, nestled at the foothills of Margalla Hills. It is in the Islamabad Capital Territory (ICT), which shares borders with the Khyber Pakhtunkhwa and Punjab provinces of Pakistan. It is the country's 9th most populated city, having a population of approximately 1.2 million. It is divided into eight different phases based on planning by the Greek architect Constantinos Apostolou Doxiadis. The area was chosen for the study due to its beauty, as it is one of the most beautiful cities in South Asia facing air pollution challenges.

Materials and methods

The collected air quality data was divided into training and testing and then analyzed by using different machine learning models. The step-by-step procedure of this study is visualized in the given Fig. 2.

Data collection

The daily air quality data ranges from 1 January 2018 to 31 August 2023 of Islamabad was collected from the Environmental Protection Agency (EPA). It consisted of NO_2, SO_2, humidity, temperature, and (Particulate matter) $PM_{2.5}$ as shown in Fig. 3. These parameters play a vital role in air quality. Temperature and Humidity affect the pollution spreading and chemical reactions. NO_2 and SO_2 are the primary sources of pollution that contribute to secondary pollution formation. $PM_{2.5}$ is riskier and responsible for mortality risks.

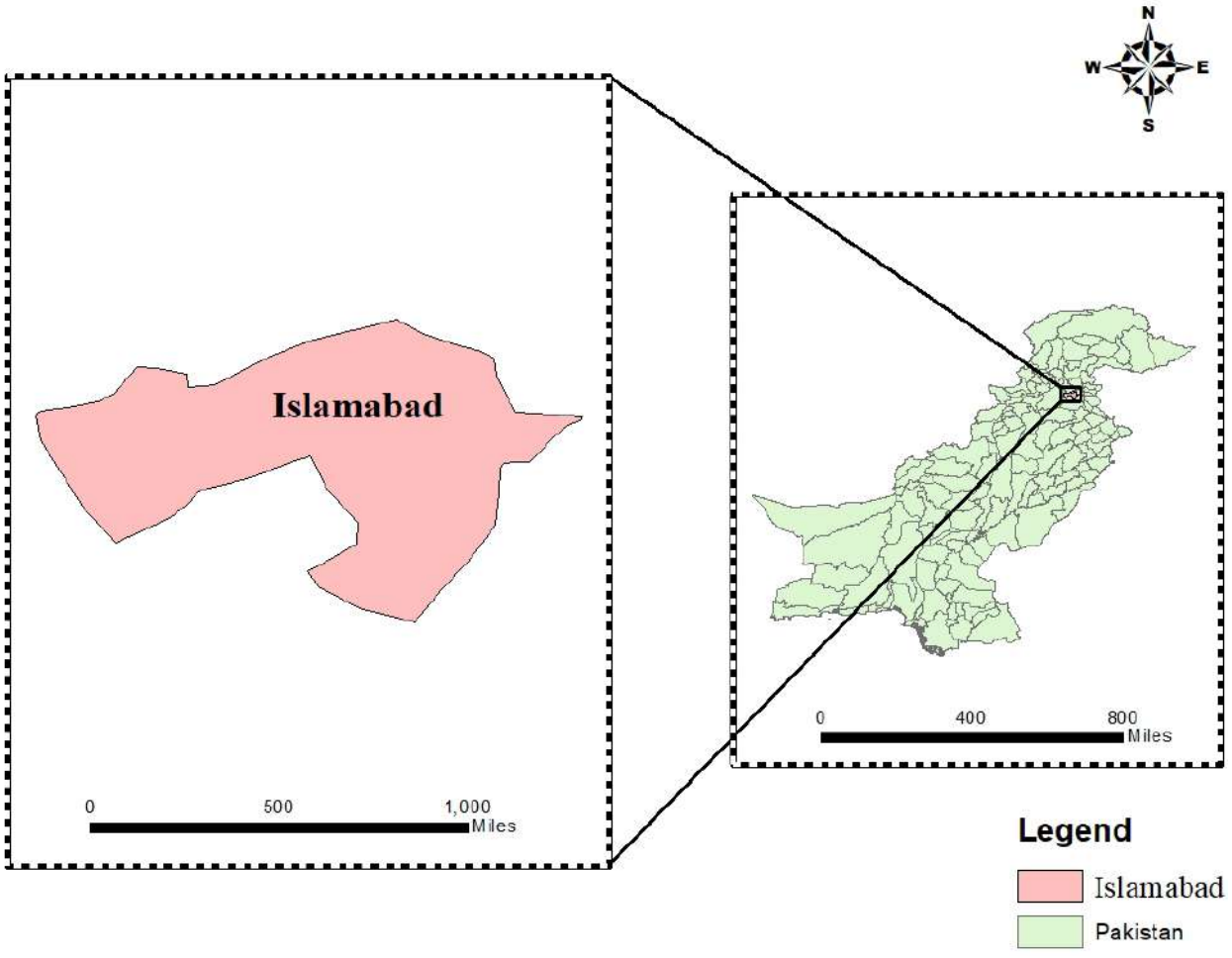


Fig. 1. Location map of study area Islamabad, Pakistan

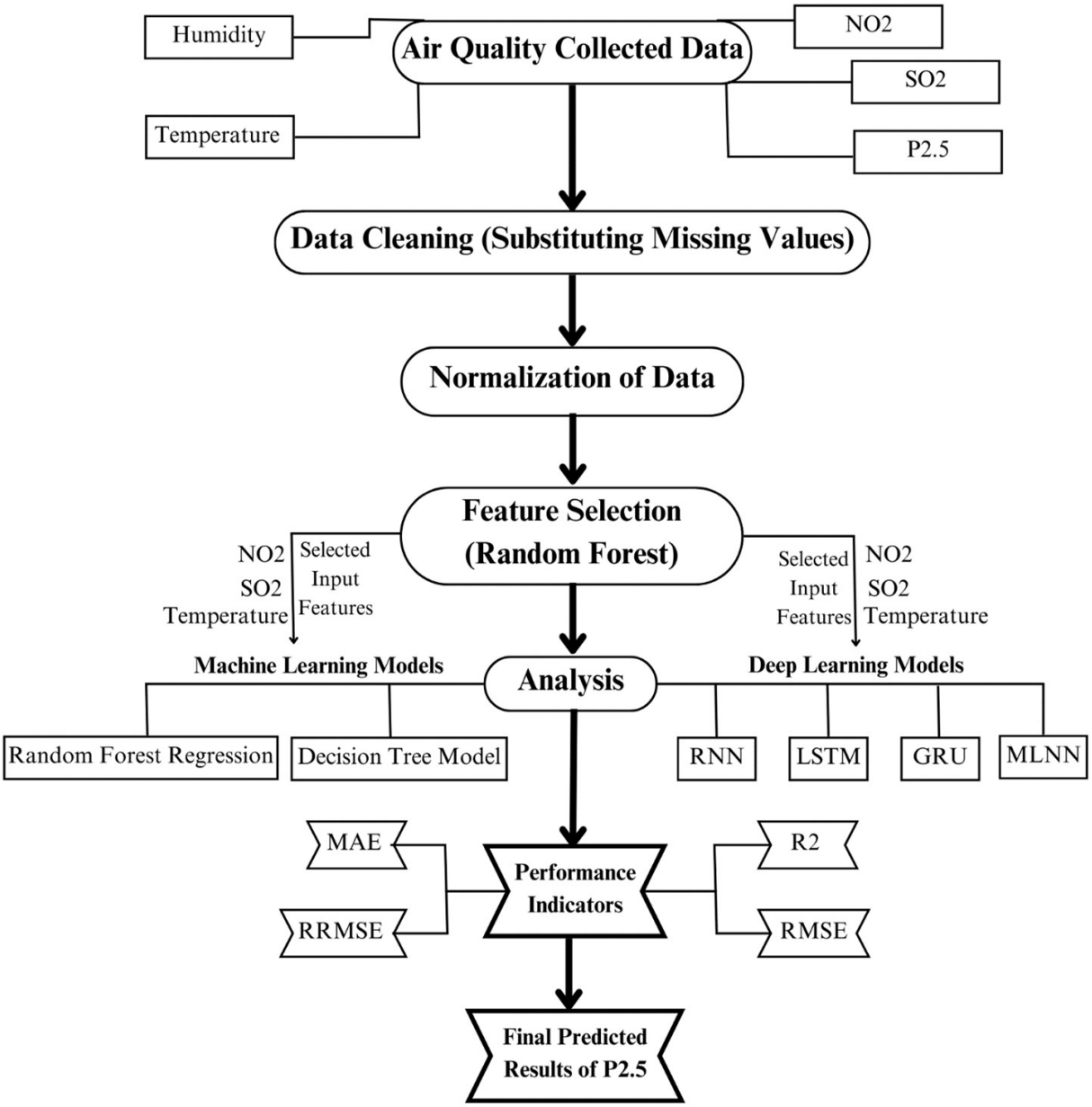


Fig. 2. Graphical representation of methodology

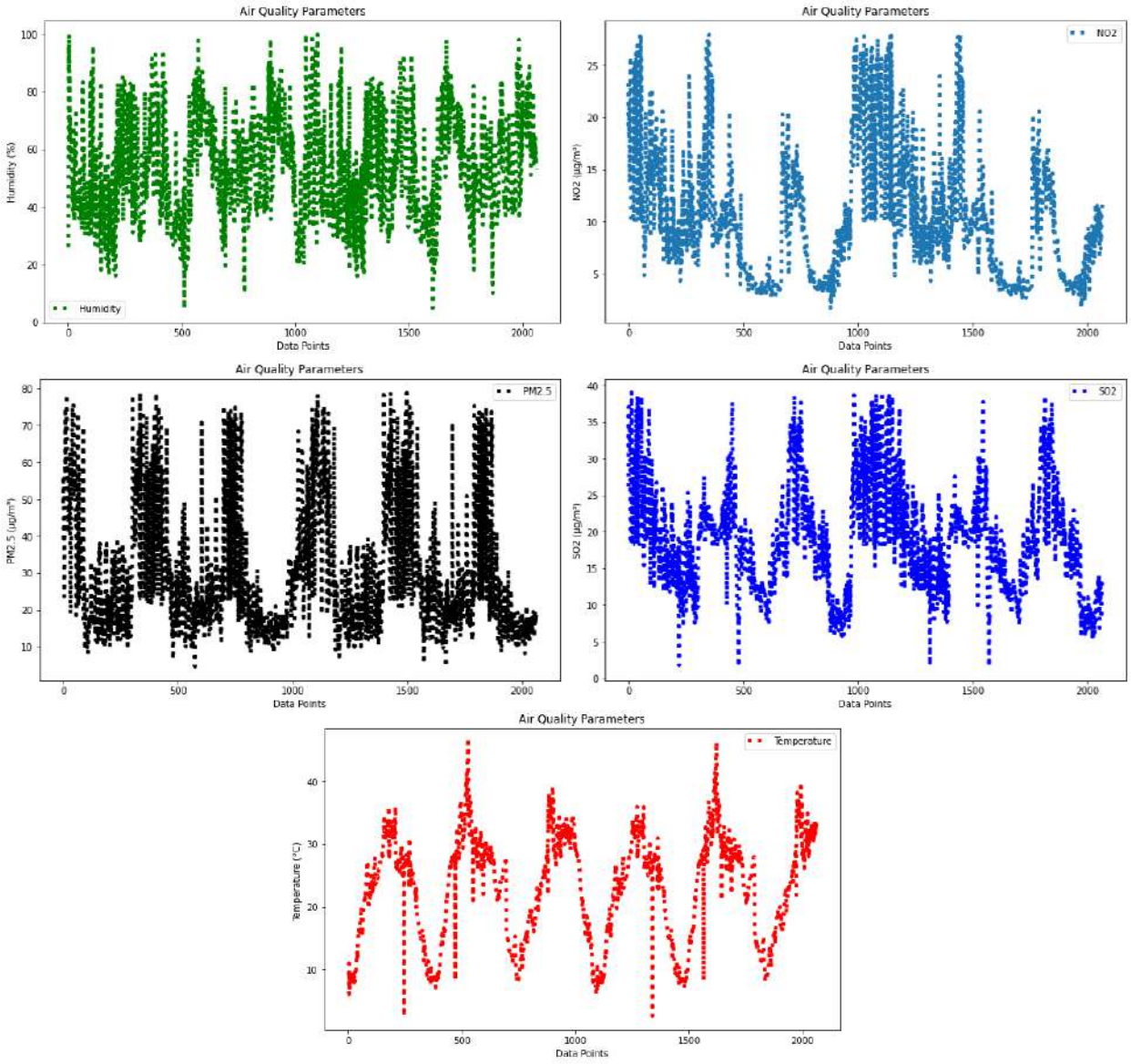


Fig. 3. Collected data containing air quality parameters

Feature engineering and selection

Feature engineering is a process used to evaluate and extract the importance of each feature in predicting a target variable. By identifying the most relevant features, this process enhances the performance and predictive ability of machine learning models. In this study, we employed the Random Forest model to assess the importance of each feature for predicting $PM_{2.5}$ concentrations. The model quantified the contribution of each feature, and based on these importance scores, we selected the most impactful features while excluding those with low relevance. The feature importance values are summarized in Table 1, with humidity being excluded due to its low contribution to model performance. Additionally, statistical analyses, including feature correlation, further support this decision by showing minimal impact of humidity on the model's predictive ability.

Table 1. Quantitative importance of different features for prediction of $PM_{2.5}$

Feature Engineering		
Name	Feature Importance values	Position of Features
Temperature	0.462086	1
SO_2	0.270166	2
NO_2	0.181008	3
Humidity	0.08674	4

Methods

The features for analysis were selected based on feature engineering. The feature importance of Humidity was found to be very low compared to other features; therefore, it was excluded from the analysis. The remaining features, including SO_2, NO_2, and temperature, were used as inputs for predicting $PM_{2.5}$. The collected data, excluding Humidity, was divided into two sets: training (80%) and testing (20%). The training data was used to train the models, while the testing data was used to assess their performance. All the models were equally optimized using a trial-and-error method. The following six models were used to predict $PM_{2.5}$ in the air of Islamabad.

Decision tree model

Decision Tree is one of the most powerful supervised machine-learning techniques used for both regression and classification analysis. It has a flow chart-like tree structure consisting of root nodes, internal nodes, branches, and leaf nodes. The internal nodes have other child nodes consisting of terms and conditions. These nodes are used to divide the data into subsets to provide further decisions. In comparison, the leaf nodes provide the results in the form of decisions. The general architecture of the Decision Tree Model is shown in Fig. 4.

In this study, to implement a Decision Tree Regressor a Python code was developed using sci-kit-learn for a regression task. It begins by splitting the dataset into training and testing sets with a 70-30 ratio, ensuring reproducibility through a fixed random state. A Decision Tree Regressor is instantiated without specifying additional parameters, and the random state is set to ensure consistent results across runs. The model is then trained using the training dataset (x_train, y_train), where it learns the underlying patterns in the data. After training, predictions are made on both the testing set (x_test) and the training set (x_train) to evaluate the model's performance. This approach allows for an assessment of how well the model generalizes to unseen data and how it performs on the training data, providing insights into potential overfitting or underfitting issues.

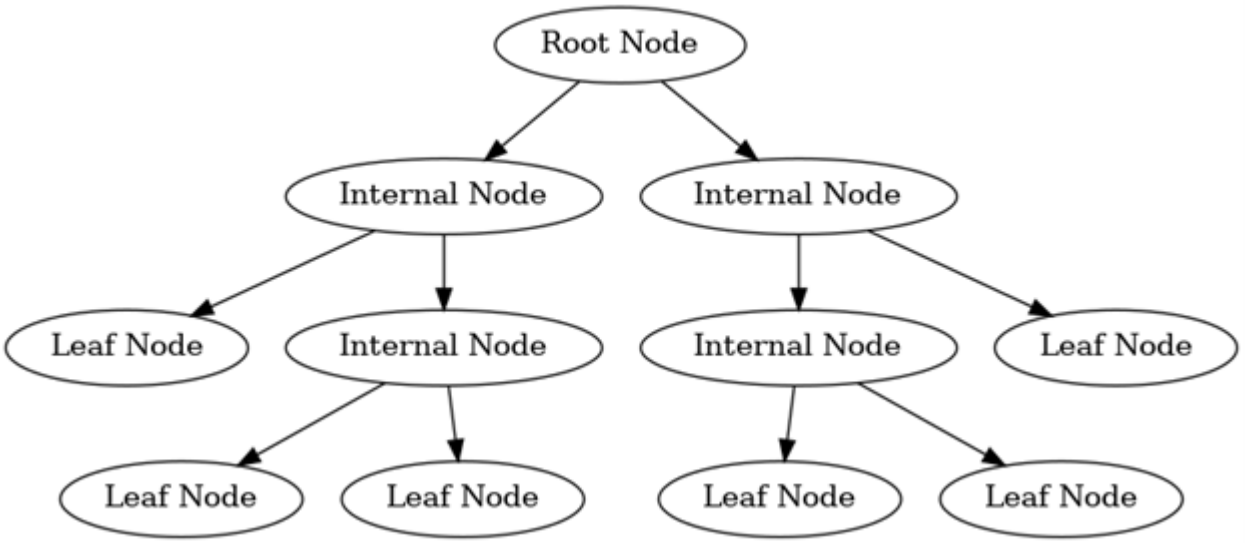


Fig. 4. The general architecture of the decision tree model

Random forest regression

Random Forest Regression combines predictions of multiple decision trees to avoid overfitting and provide accurate results. The model depends on two main parameters including several predictors and many grown trees. In 1st stage, the model begins with several trees and then it provides a classification tree for each predictor, randomly. During predictions, the results of all the decision trees in the forest are averaged to obtain accuracy. During the growing tree process, this model uses extra randomness instead of searching for the best feature among the random subset of features. It gains higher bias for lower variance yielding generally a better result. In the case of regression, it mainly depends on the average results of all trees in the model. Mathematically it can be shown as:

$$\acute{y} = \frac{1}{T}(\sum_{t=1}^{T} \acute{y}_t(x) \quad (1)$$

ý = final prediction of RF model for an input x.

T = number of decision trees in the forest.

$\acute{y}_t$ (x) = prediction from t-th trees for an input x.

By using different inputs, the model creates a decision tree for each input representing possible outcomes. Then for each input, it provides a prediction $\acute{y}_t$ (x). The final prediction ý is the average of all the predictions from T-trees. The general architectural view of the whole process is illustrated in Fig. 5.

In this study, python code was developed to implement a Random Forest Regressor for regression using sci-kit-learn, designed to predict target values based on input features. It begins by splitting the dataset into training and testing sets in an 80-20 ratio, ensuring reproducibility with a fixed random state. The Random Forest Regressor is initialized with 2000 decision trees and uses the mean squared error as the criterion for evaluating split quality. The model is then trained on the training dataset, learning patterns, and relationships between the features and target values. After training, predictions are made on both the training and testing sets to assess the model's performance. The high number of estimators improves accuracy but also increases computational requirements, demonstrating a balance between achieving high predictive accuracy and managing computational resources.

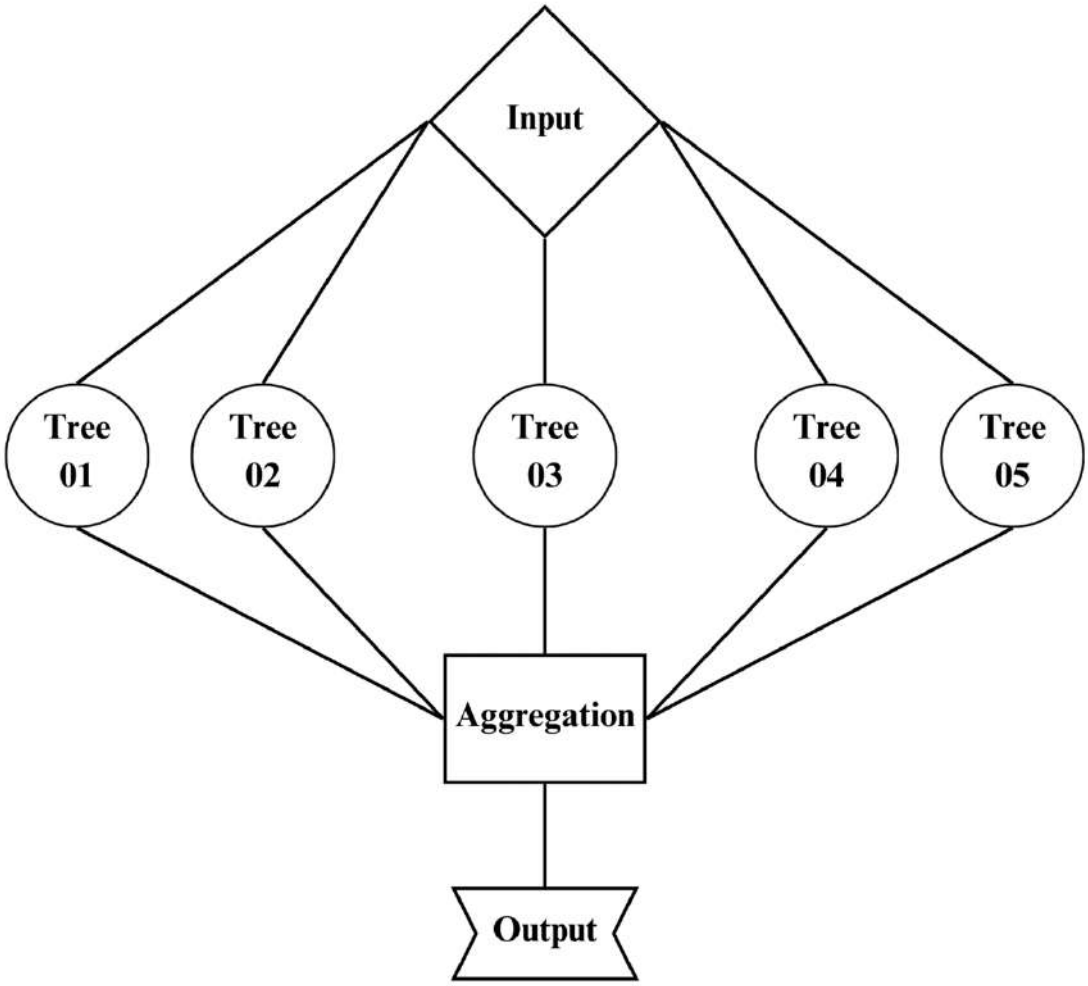


Fig. 5. Random forest model architecture

Recurrent neural network (RNN)

RNN is a type of Artificial Neural Network (ANN) having an input layer, hidden layers, and output layers. The hidden layers extract information from the previous points in a sequence, that is why it is well-suited for the time series data. It processes the data as a cycle by maintaining the memory of previous inputs. This is mainly used for speech recognition, natural language processing, etc. In the 1st phase of analysis, the input data is used for prediction in the model, then it moves toward hidden state analysis [15]. In the hidden state, a hidden layer of neural networks is used to capture the data accurately and then provide it as output. The architecture of RNN is shown in Fig. 6. Mathematical computation output from the RNN model is:

$$y_t = g(W_{hy} b_t + b_y) \quad (2)$$

y_t = output of the model.

g = activation function of the model.

W_{hy} = Weight function for the hidden state to output.

b_y = bias term of the model.

Similarly, the mathematical equation for its

hidden states is given by.

$$ht = \sigma(W_{xt} + Uh_{t-1} + b_h) \quad (3)$$

W_{xt} = Transformation of current input xt.

Uh_{t-1} = Transformation of the previous hidden state h_{t-1}.

b_h = Bias vector for the hidden state.

σ = Activation function (ReLU function).

Its architecture is shown in Fig. 6.

In this research, python code was developed to implement RNN using TensorFlow and Keras for a regression task, with a focus on time-series or sequential data. It begins by normalizing the output (`y`) using `MinMaxScaler` to scale values between 0 and 1, ensuring consistent output ranges for efficient learning. The dataset is then split into training and testing sets with an 80-20 ratio, using a fixed random state for reproducibility. The input features (`X`) are standardized using `StandardScaler` to achieve a mean of 0 and a standard deviation of 1, which is crucial for enhancing the model's performance. The standardized data is reshaped into a 3D array (`samples, time steps, features`) suitable for RNN input. The RNN model is built with a single Simple RNN layer containing 50 units and using the sigmoid activation function, followed by a dense layer with a single neuron for the regression output. The model is compiled using the Adam optimizer and mean squared error (MSE) as the loss function. It is trained for 300 epochs with a batch size of 64, optimizing the model's parameters to minimize prediction errors.

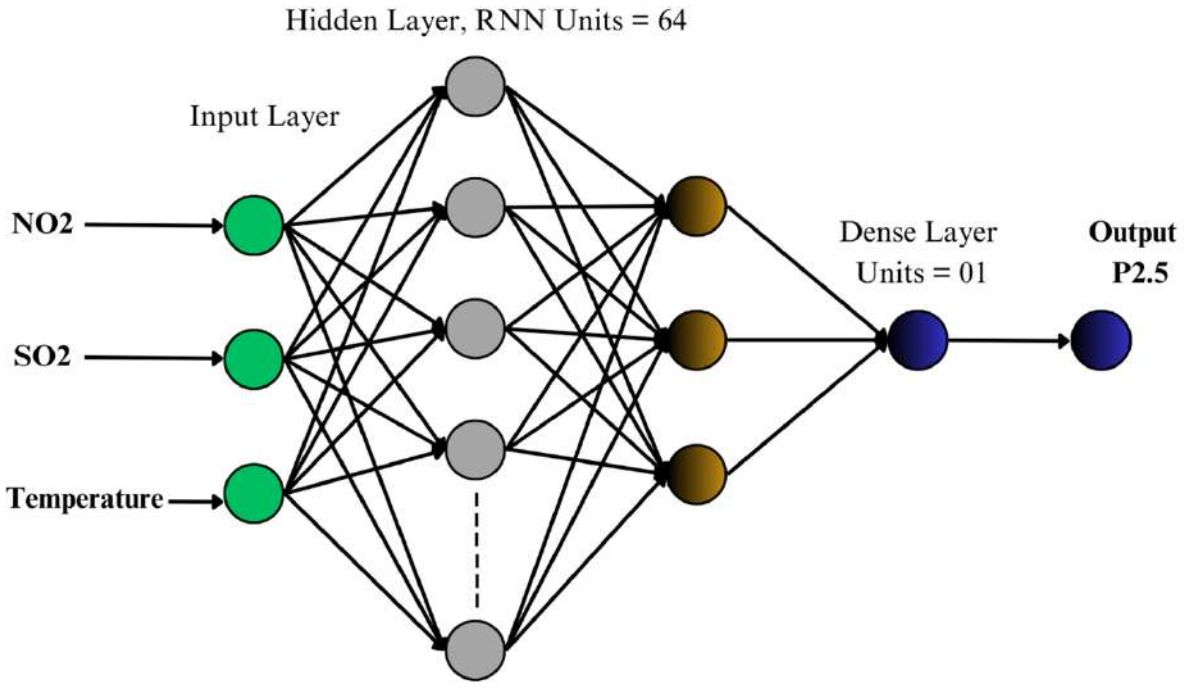


Fig. 6. Illustrates the architecture of the RNN model

Long short-term memory (LSTM)

LSTM is an improved version of the RNN model having the ability to capture long-term dependencies in sequential data. It is well-suited for sequential data. It has three gates including the input gate, forget gate, and output gate. The input gate controls the information added to the memory cell, the output gate controls the output information from the memory cell while the forget gate controls the information that is no longer useful in the memory cell. The algorithm not only uses current input but also keeps the previous results in memory. Fig. 7 shows the architecture of the LSTM model. The mathematical expression is divided into 3 parts including input gate, forget gate, and output gate.

Input gate [16]:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

W_i = weight matrix for the input gate.

b_i = bias term for input gate.

Forget gate:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (5)$$

W_f = weight matrix of forget gate.

b_f = bias term for forget gate.

σ = activation function for forgetting gate.

The mathematical equation for the last stage of the output gate is.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (6)$$

W_o = Matric weight for its output gate.

b_o = bias term for output gate.

σ = activation function for output gate.

In this study, Python code was constructed for Long Short-Term Memory (LSTM) neural network development for a regression task using TensorFlow and Keras library. It starts by normalizing the target variable (y) with MinMaxScaler to fit it within a 0-1 range, essential for consistent and effective learning. The data is split into training and testing sets with 70% for training and 30% for testing, ensuring reproducibility with a fixed random state. The input features (X) are standardized using Standard Scaler to have a mean of 0 and a standard deviation of 1, enhancing model performance. The input data is then reshaped to a 3D format (samples, time steps, features), suitable for LSTM networks that require sequential input. The LSTM model is constructed with 60 units and a ReLU activation function, followed by a dense layer with a single neuron for regression output. Compiled using the Adam optimizer and Mean Squared Error (MSE) loss function, the model is trained for 3000 epochs with a batch size of 16, optimizing the weights to improve the model's predictive accuracy.

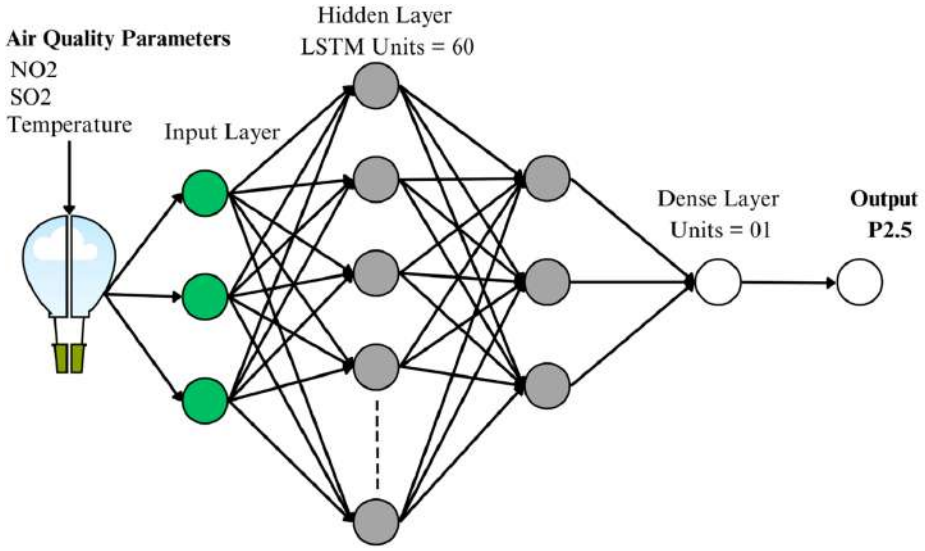


Fig. 7. Illustrates the architecture of the LSTM model

Gated recurrent unit (GRU)

GRU is also a type of RNN model that is an alternative to the LSTM model and can be used for text, speech, and time series data just like LSTM. It is different from LSTM due to its gated mechanism. It has two gates, a reset gate and an update gate. The reset gate is used to control the amount of information that should be forgotten while the update gate is used to control the newly added information for updating the hidden state of the model. The architecture of the GRU model is illustrated in Fig. 8. Mathematically updated gate can be expressed as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (7)\ [17]$$

$W_z x_t$ = transformation of the current input x_t.

$U_z h_{t-1}$= transformation of the previously hidden state h_{t-1}.

b_z = bias vector for the update gate.

σ = activation function.

Reset gate:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (8)$$

h_{t-1} = hidden state vector from the previous state.

W_r = weight of input matrix.

b_r = bias vector for reset gate.

In this study, python code was constructed to train a Simple RNN using TensorFlow and Keras for a regression problem. It begins by normalizing the output data (`y`) with `MinMaxScaler`, transforming it into a range between 0 and 1 for better performance during training. The dataset is split into training and testing sets in an 80-20 ratio, with a fixed random state to ensure reproducibility. The input features (`X`) are then standardized using `StandardScaler`, which adjusts the data to have a mean of 0 and a standard deviation of 1, facilitating faster convergence during training. The standardized input is reshaped into a 3D array to conform to the RNN input requirements, specifying dimensions for samples, time steps, and features. A Simple RNN model is constructed with 600 units and a ReLU activation function, followed by a dense layer with a single neuron for the regression output. The model is compiled using the Adam optimizer and mean squared error (MSE) as the loss function. It is trained for 10,000 epochs with a batch size of 128, optimizing the network's weights to minimize prediction errors and enhance its predictive capabilities.

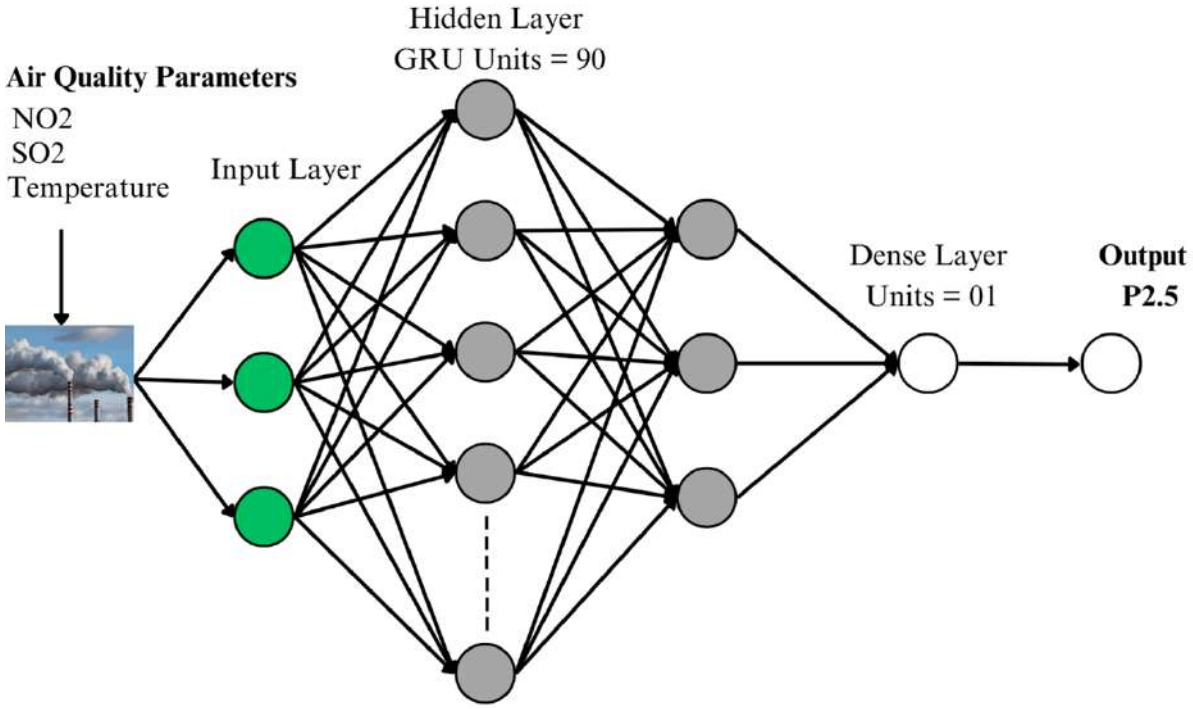


Fig. 8. Illustrate the architecture of the GRU model

Multi-Layer neural network

MLNN is also known as Multi-Layer Perception (MLP) and is commonly used for continuous data patterns. It is a typical example of a feedforward neural network and consists of input, hidden, and output layers. The input layer has nodes or neurons having features and dimensions. The input neurons may be different depending on the number of inputs. Similarly, the number of neurons in the output may be single or multiple. A backpropagation algorithm trains the model. Fig. 9 explains the architecture of the MLNN model. Mathematically it is divided into two parts, hidden state and output state [18].

Hidden state

$$z^{(I)} = W^{(I)}a^{(I-1)} + b^{(I)} \quad (9)$$

$$a^{(I)} = f\left(z^{(I)}\right) \quad (10)$$

$W^{(I)}$ = weight matrix for layer I.

$a^{(I-1)}$ = activation output from the previous layer.

$b^{(I)}$ = bias vector for layer I.

f = activation function (Relu).

$z^{(I)}$ = weighted input to layer I.

Output Layer:

$$z^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)} \quad (11)$$

$$a^{(L)} = f\left(z^{(L)}\right) \quad (12)$$

$W^{(L)}$ = weight matrix for the output layer.

$a^{(I-1)}$ = activation output from the last hidden layer.

$b^{(I)}$ = bias vector for the output layer.

f = activation function (Relu).

$z^{(I)}$ = weighted input to output.

In this study, the output data (y) data was normalized using MinMaxScaler and then the dataset was split into training and testing sets to ensure reproducibility. The input features (X) are standardized with StandardScaler to maintain consistent feature scaling, crucial for efficient model training. The MLNN model is then built with three hidden layers (128, 64, and 32 neurons) using the ReLU activation function, and a single neuron in the output layer for the regression output. Compiled with the Adam optimizer and Mean Squared Error (MSE) as the loss function, the model is trained for 7000 epochs with a size of 64, optimizing its parameters to minimize prediction error and improve accuracy.

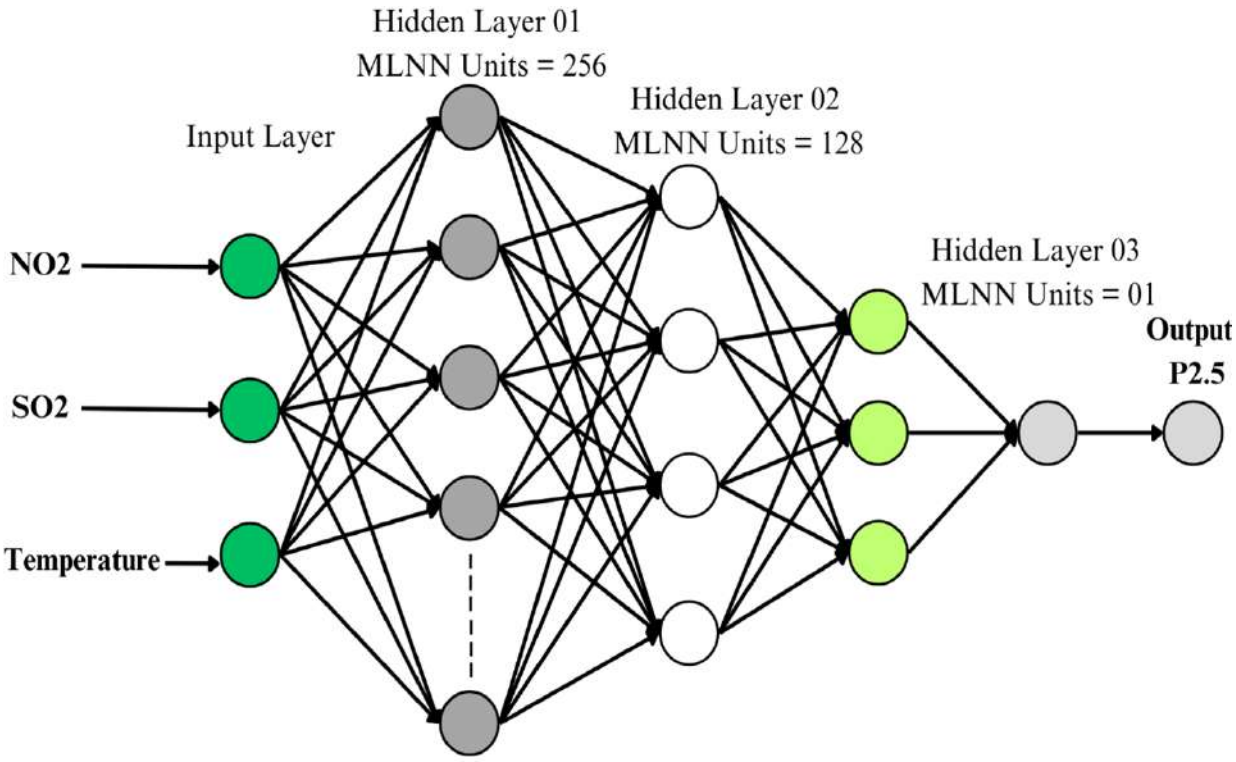


Fig. 9. Illustrate the architecture of the MLNN model

Model performance indicators

Four statistical indicators including R-square, Root Mean Square Error (RMSE), Relative Root Mean Square Error (RRMSE) and Mean Absolute Error (MAE) are used to assess the performance of these models. These indicators made it easy to decide which model is best.

Coefficient of determination (R^2)

R^2 measures how well the model's predictions match the actual observed values, ranging from 0 to 1, where a value closer to 1 indicates better prediction accuracy. This helps in understanding how well the model captures the underlying patterns in the data. In the context of $PM_{2.5}$ prediction, a high R^2 indicates that the model can accurately predict air quality levels based on input features, which is crucial for environmental monitoring and decision-making. Mathematically R^2 can be expressed as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(yi-\acute{y}i)^2}{\sum_{i=1}^{n}(yi-ym)^2} \quad (13)$$

Here yi = Observed value of the target variable.

ýi = predicted value of the target variable.

ym = mean value of the target variable.

Root mean square error (RMSE)

RMSE is a common measure used to evaluate the difference between observed and predicted values. It has the advantage of penalizing larger errors, which is important in applications like air quality prediction, where large deviations from the true value could have significant consequences. RMSE gives an intuitive measure of prediction accuracy in the same units as the target variable ($PM_{2.5}$ concentration), making it easy to interpret for policymakers and environmental agencies. A mathematical equation for finding RMSE is shown.

$$RMSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(yi-\acute{y}i)^2} \quad (14)$$

yi = Observed value of the target variable.

ýi = predicted value of the target variable.

Relative root mean square error (RRMSE)

RRMSE normalizes RMSE by the range of observed values, making it easier to compare model performance across different datasets. For $PM_{2.5}$ prediction, RRMSE helps assess the model's ability to generalize across various pollution levels, which is important in regions with fluctuating air quality. This can be calculated by:

$$RRMSE = \frac{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(yi-\acute{y}i)^2}}{(ymax - ymin)} \quad (15)$$

yi = Observed value of the target variable.

ýi = predicted value of the target variable.

ymax = maximum value of the target variable.

ymin = minimum value of the target variable.

Mean absolute error (MAE)

MAE measures the average absolute difference between observed and predicted values. Unlike RMSE, it does not penalize large errors as heavily, but it provides a straightforward interpretation of prediction accuracy. MAE is particularly useful for understanding the magnitude of typical prediction errors, helping to quantify how close the predicted $PM_{2.5}$ values are to actual observed levels. The mathematical equation for MAE is given as:

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}|yi - \acute{y}| \quad (16)$$

yi = Observed value of the target variable.

ýi = predicted value of the target variable.

Compromise programming (CP)

CP is a well-established technique that allows for the ranking of statistical models based on their performance, considering multiple criteria simultaneously. In this study, CP was employed to rank the machine learning models used for $PM_{2.5}$ prediction based on key performance indicators such as R^2, RMSE, RRMSE, and MAE.

The LP metric is used to quantify the "distance" between the actual model performance values and the ideal values (where the model perfectly predicts the target). The formula for calculating the LP metric is as follows:

$$Lp = [\sum_{i=1}^{n} |Wn * - Wn|^m]^{\frac{1}{m}} \quad (17)$$

Where:

- W_n* is the observed value of the statistical performance measure for model nn.
- W_n is the ideal value of the performance measure (i.e., the perfect value, representing a scenario where the model's prediction perfectly matches the observed data).
- mm is a parameter (typically m=1m = 1 or m=2m = 2) that controls the sensitivity of the metric to deviations from the ideal value.

Results and discussion

In this research, the performance of machine learning and deep learning models was compared on particulate matter $PM_{2.5}$ prediction. The collected data was divided into training and testing sets with a ratio of 80:20. This data was analyzed by using machine and deep learning models. All the models performed best in predicting the air quality. Machine learning models performed better in only training, but deep learning models were better in both training and testing. The results of all these models are shown in Table 2 and the heatmap.

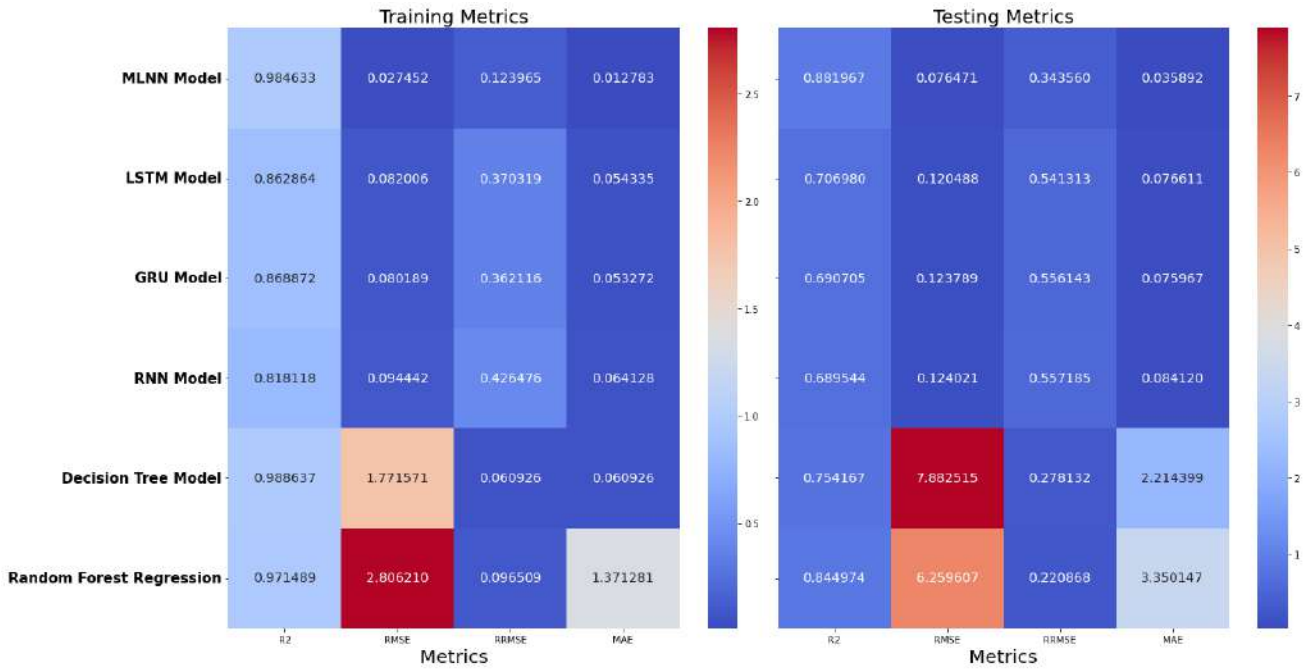


Fig. 10. Representation of model performance metrics by Heatmap

Evaluation of decision tree model

Fig. 10 demonstrates the results of all the models. It is clear from the results that the decision tree model is best in the training by achieving the higher R^2 value of 0.98 and lower RMSE (1.7716) and MAE (0.0609) values. This higher R^2 value shows that the model is closely fit to the data pattern and precisely predicts the data in training. However, the performance of the Decision Tree model is not much better in testing as compared to the training. The Decision Tree model failed to maintain their superior's position in the testing phase. It attained lower R^2 values and higher RMSE and RRMSE values in the validation phase as compared to training, suggesting poor performance in the testing. This poor performance of DT model in the testing phase is due to its rigid structure of multiple trees and instability. Fig. 11 (A) shows the actual data in comparison with the predicted data Decision Tree model. The observed and predicted data are differentiated by using distinct colors of lines. A vertical boundary line is provided in plots for separating the training and testing parts of data. The predicted lines are close to the observed values only in the training set and have some more gap than the other models showing the shallow ability of the Decision Tree in the testing phase.

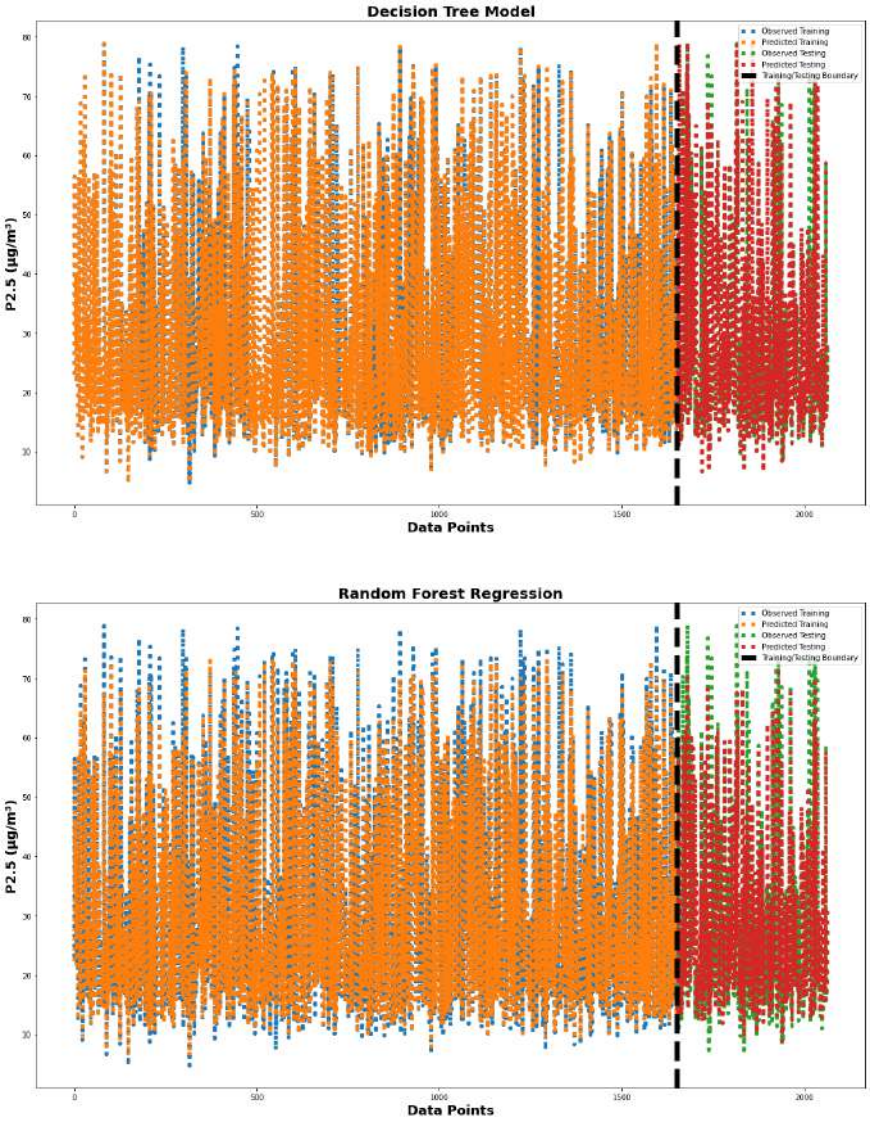
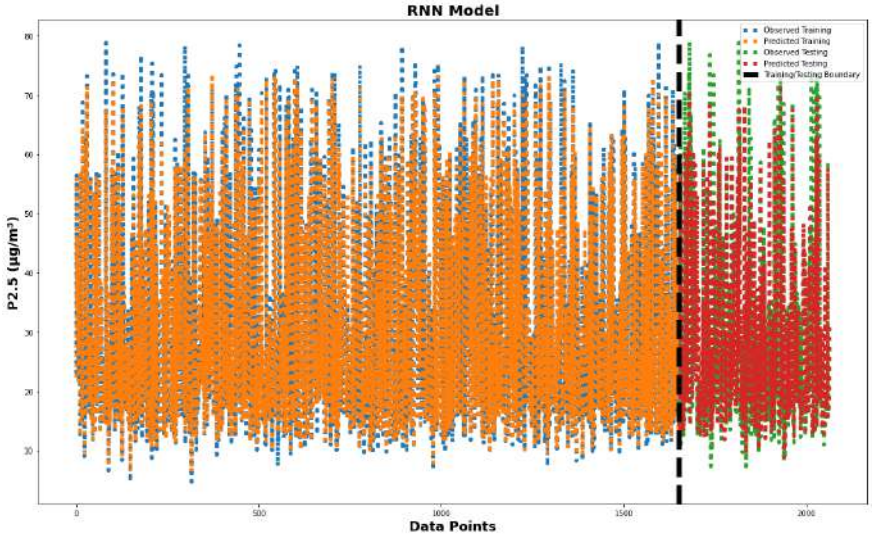


Fig. 11. Actual vs predicted $PM_{2.5}$ values for training and testing phases by decision tree model and random forest

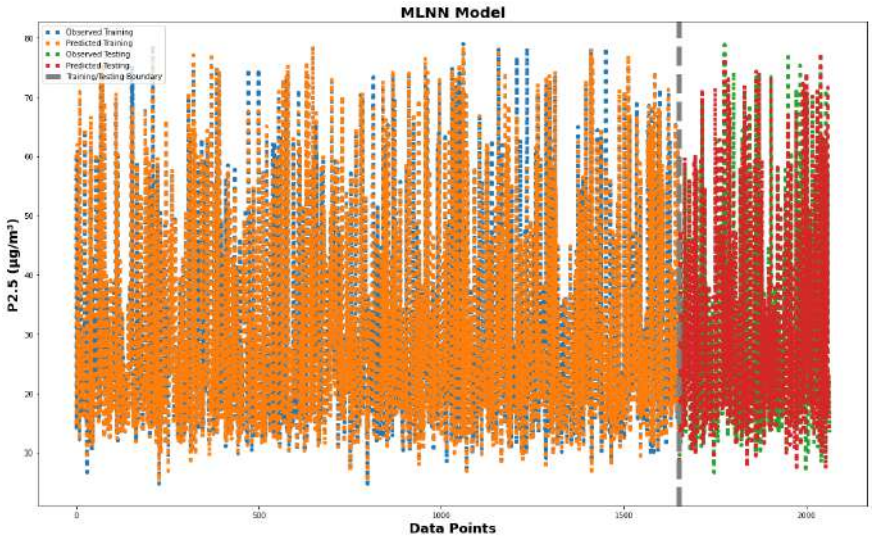
Evaluation of random forest model

Similarly, Random Forest is better in training and has a higher R^2 value of 0.97. However, the performance of RF models is not much better in testing than in training. RF model failed to maintain their superior's position in the testing phase. Due to high variance in the data and less hypermeters it attained lower R^2 values and higher RMSE and RRMSE values in the validation phase as compared to training, suggesting poor performance in the testing. Fig. 11 (B) shows the predicting abilities of the Random Forest model. The predicted lines are close to the observed values only in the training set and have some more gap than the other models showing the shallow ability of the Decision Tree in the testing phase.

(A)



(B)



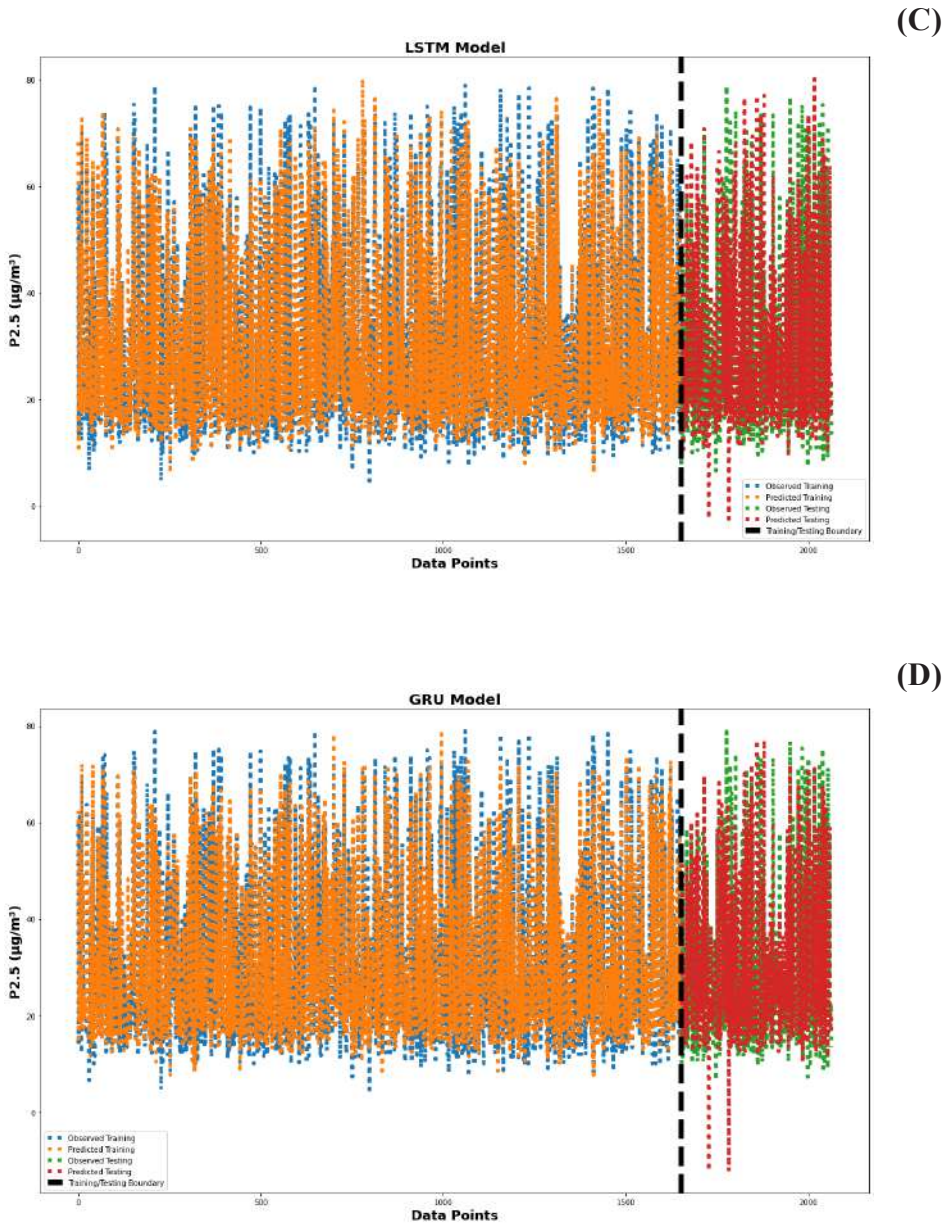


Fig. 12. Actual vs pedicted $PM_{2.5}$ Values for training and testing phases by (A) RNN model (B) MLNN model (C) LSTM model (D) GRU model

Evaluation of RNNs

RNNs are a class of neural networks designed to deal the sequential data. In this study, the RNN model maintained a solid performance both in training and testing with the Adam optimizer having a learning rate of 0.001 and the Min-Max scaling technique [19]. The best results of RNN in both training and testing indicate its better-predicting power. It acquired the best R^2 value of 0.81 in training followed by an R^2 value of 0.68 in testing. Similarly, it accomplished lower error (RMSE, RRMSE, MAE) values. These best results of the RNN model suggest that this

model has the capability of capturing complex time series. Fig. 12 (A) shows the predicted and actual data by the RNN Model. The predicted and observed lines are also close to each other but have some minor gaps showing moderate capturing abilities of the GRU and RNN models.

Evaluation of multi-layers neural network

The Multi-Layers Neural Network (MLNN) model demonstrated strong performance in predicting $PM_{2.5}$ in the air with an R^2 value of 0.98 in training and 0.88 in the testing phase. These higher R^2 values and lower error rates indicate good prediction accuracy of this model. The model attained these results even on low epochs with a learning rate of 0.001. Because MLNN model process all the input at once and does not need to consider temporal sequence dependencies. Fig. 12 (B) the actual and predicted data by the MLNN model. In this Figure the predicted training lines are just above the observed data lines, indicating an accurate prediction of the MLNN model.

Evaluation of the LSTM model

The LSTM model performed comparatively better than machine learning models. At the first stage, training this model was difficult, because it attained the best results on the epochs of more than 1000. Due to week temporal relationship between the input data and $PM_{2.5}$, LSTM model unperformed as compared to the MLNN model. But despite all this, the model performed best in both the training and testing with higher R^2 values of 0.86, and 0.70 and lower RMSE, RRMSE, and MAE values. The analysis through the model was performed by using a technique of Min-Max scaling and a learning rate of 0.001 with an Adam optimizer. Fig. 12 (C) represents the actual and predicted data by the LSTM model. In this figure the predicted training lines are just above the observed data lines, indicating an accurate prediction of the LSTM model.

Evaluation of GRU model

As GRU model is an alternative to the LSTM and performed better in both the training and testing phases. The training R^2 value of the GRU model was approximately equal to that of LSTM but in the testing phase R^2 value of GRU was lower than LSTM. The training process of the GRU model was not difficult and the best results were obtained on lower epochs. Fig. 12 (D) shows the predicted and actual data by the GRU Model. The predicted and observed lines are also close to each other but have some minor gaps showing moderate capturing abilities of the GRU model.

Ranking of models via compromise programming

These models cannot be ranked only based on a single statistical indicator, because some models have higher R^2 values which is evidence of the best performance, but the same models have also higher error matrices leading towards weak performance. Therefore, all these models are ranked not only based on a single statistical metric but on the combined effect of these metrics. A compromise programming technique is used to get the combined effect of each model. Compromise programming results revealed that due to better performance in both training and testing the MLNN model was ranked at the top position. Whereas LSTM, GRU, RNN, Decision Tree, and Random Forest were placed at 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, and 6^{th} position, respectively. The top position of the MLNN model suggests that the MLNN model is the best model for predicting the particulate matter $PM_{2.5}$ in the air quality of Islamabad Pakistan. The ranking results of these models are shown in Table 2.

Table 2. The ranking of models

Models	Training				Testing				Ranking
	R-Square	RMSE	RRMSE	MAE	R-Square	RMSE	RRMSE	MAE	
MLNN	0.985	0.027	0.124	0.013	0.882	0.076	0.344	0.036	1
LSTM	0.863	0.082	0.37	0.054	0.707	0.12	0.541	0.077	2
GRU	0.869	0.08	0.362	0.053	0.691	0.124	0.556	0.076	3
RNN	0.818	0.094	0.426	0.064	0.69	0.124	0.557	0.084	4
Decision Tree	0.989	1.772	0.061	0.061	0.754	7.883	0.278	2.214	5
Random Forest	0.971	2.806	0.097	1.371	0.845	6.26	0.221	3.35	6

The research study provided a comparison of machine learning and deep learning models in the prediction of particulate matter $PM_{2.5}$ in air. The study used NO_2, SO_2, and temperature as input for prediction because these constituents are deeply linked with $PM_{2.5}$. It enhanced the prediction accuracy of particulate matter $PM_{2.5}$ in the air of Islamabad and provided accurate results for better air quality management and prediction. Due to poor capturing capability of complex data and noise fitting nature of machine learning models, they could not achieve best results, especially in the testing phase [20].

The Decision Tree model is sensitive to a small change in the input data leading towards inconsistency in predictions [21]. While Random Forest has multiple trees but still fails in capturing complex data. Therefore, due to the time series complex data these models could not achieve the data pattern in the testing phase and compromised on the predictive accuracy of air quality prediction. They provided accurate results only in the training phase up to R^2 0.98 and 0.97.

On the other hand, deep learning models are highly able to capture complex and time series data. They have the capability of automatic feature extraction and are designed for capturing non-linear data due to their multiple layers [22, 23]. RNN model is designed for sequential data but faces vanishing gradient problems in complex and long-term data. This situation makes the model less effective in maintaining long-term memory. Here it captured the data accurately in both phases but could not achieve superior position as compared to LSTM, MLNN, and GRU. The GRU model is an alternative to LSTM model and has the capability of capturing time-series data. But here due to the complex nature of data, it could not gain the relationship more correctly as compared to LSTM and MLNN models. LSTM model has capability of long-term memory and does not face vanishing gradient problems due to its gated mechanism nature. But it still can suffer from computational inefficiencies. Here LSTM is underperformed when compared to MLNN model. The MLNN model has different layers with different activation functions. It has a sequential data pattern capturing capacity. That is why the results of this model were best of all the other models. The model achieved a higher accuracy of 0.98 R^2 in training with 0.88 in the testing phase. These results of the deep learning models, especially of MLNN, suggest their best prediction capability. Here deep learning models gained the pattern of data correctly in both phases training and validation and therefore, these models are proved to be the top-ranking in prediction of particulate matter $PM_{2.5}$ in the air of Islamabad, Pakistan.

Study limitations

This study utilized only NO_2, SO_2, and temperature as input for predicting $PM_{2.5}$, because these variables are directly involved in the $PM_{2.5}$ formation and variation and ignorance of these elements may limit the model ability to capture the data. The other atmospheric variables including precipitation, wind speed etc. are also involved in the $PM_{2.5}$ variation but exclusion of these variables is the limitation of this study. Because these variables can lead towards dispersion and chemical transformation of $PM_{2.5}$. However, future studies can include multiple environmental and metrological factors to enhance these models' prediction power. Furthermore, the study is conducted for Islamabad city having dense urbanization, vehicular emissions, and temperature variation. It is applicable to the regions having similar $PM_{2.5}$ pollution sources and climate variation as Islamabad. But its generalizability to other regions will require further investigation. For regions with different environmental conditions and different $PM_{2.5}$ pollution sources, modification to model input parameters and structure will be necessary. The performance and ranking pattern of these models can also be changed by changing the input parameters.

Practical implications of study

The research work has important implications for air quality management. Industries and power plants as major sources of NO_2, and SO_2 can utilize this for regulating and reducing pollution. By identifying the primary contributor of $PM_{2.5}$ from the input data, the policymakers can reduce their fast dispersion to secure the healthy environment. Furthermore, the relation between temperature and $PM_{2.5}$ can provide timely warning about rise in $PM_{2.5}$ level in the air. It can also be used in preparing air quality management policies.

Conclusion

In this study, six machine learning models, including four neural network architectures, were utilized to predict the air quality of Islamabad, Pakistan. These models were MLNN, LSTM, GRU, RNN, Random Forest, and Decision Tree. The input features included temperature, SO_2, and NO_2, while $PM_{2.5}$ was the target variable. The performance of these models was evaluated using statistical indicators such as R^2, RMSE, RRMSE, and MAE. Additionally, a compromise programming technique was employed to rank the models based on their overall performance. The results revealed that the MLNN model outperformed the others, achieving the highest R^2 value of 0.98 and the lowest RMSE, RRMSE, and MAE values during both the training and testing phases. Other models were ranked as follows: LSTM (2nd), RNN (3rd), GRU (4^{th}), Decision Tree (5^{th}), and Random Forest (6^{th}).

Despite the MLNN model's robust performance, its limitations must be acknowledged. Like all machine learning models, MLNN's accuracy is dependent on the quality and quantity of data, as well as the selection of features and hyperparameters. In this study, only three input features (temperature, SO_2, NO_2) were used, which may not capture the full complexity of $PM_{2.5}$ variations influenced by other meteorological and environmental factors. Moreover, while compromise programming provided an objective ranking, it is important to consider the practical feasibility of deploying these models for real-time air quality monitoring and management.

To further enhance $PM_{2.5}$ prediction, future studies should explore incorporating additional features such as wind speed, humidity, and vehicular emissions to better capture the dynamic factors influencing air quality. Hyperparameter tuning should be performed systematically using advanced optimization techniques like grid search, random search, or Bayesian optimization to identify the optimal configurations for each model. Additionally, alternative modeling approaches, such as ensemble learning or hybrid models, could be investigated to improve predictive accuracy and robustness.

For practical implementation, it is crucial to integrate predictive models into air quality monitoring systems and decision-support frameworks to provide actionable insights for policymakers and stakeholders. This research provides a foundation for developing more accurate and reliable $PM_{2.5}$ prediction tools, contributing to air pollution mitigation and public health protection. A sustainable and clean environment can be achieved by combining improved models with proactive air quality management strategies.

Financial supports

The research study has no funding sources, and no fund or grant was received during the preparation of the manuscript.

Competing interests

The authors declare no competing interests.

Author's contributions

Muhammad Waqas: conceptualization, prepared Machine & Deep Learning models Python codes, analyzed the data properly, and wrote the manuscript. Responsible for correspondence, submission, and revision of the paper.

Shahid Noor Jan: Visualized the collected data and results by using several types of plots. The study area map was also prepared.

Basir Ullah: Conceptualization, revision of manuscript and guidance in analysis.

Afed Ullah Khan: The Authors performed the tasks including data collection, data cleaning, guidance in the analysis, supervision of writing, and revision of the final manuscript.

Ateeq Ur Rauf: Editing manuscript, supervision, reviewing the manuscript, guidance in the analysis process.

Bakht Niaz: Conceptualization, supervision, reviewing, and editing the manuscript.

All the authors have read, understood, and decided to submit the manuscript.

Acknowledgements

The authors sincerely appreciate the Environmental Protection Agency (EPA) for generously providing the data used in this research.

Ethical considerations

"Ethical issues (Including plagiarism, Informed Consent, misconduct, data fabrication and/or falsification, double publication and/or submission, redundancy, etc) have been completely ob-served by the authors."

References

1. Gul H, Das B. The Impacts of Air Pollution on Human Health and Well-Being: A Comprehensive Review. Journal of Environmental Impact and Management Policy. 2023:1-11.

2. Wang C, Miao Z, Chen X, Cheng Y. Factors affecting changes of greenhouse gas emissions in Belt and Road countries. Renewable and Sustainable Energy Reviews. 2021;147:111220.

3. Wang Q, Liu S. The Effects and Pathogenesis of $PM_{2.5}$ and Its Components on Chronic Obstructive Pulmonary Disease. Int J Chron Obstruct Pulmon Dis. 2023;18:493-506.

4. Tran HM, Tsai F-J, Lee Y-L, Chang J-H, Chang L-T, Chang T-Y, et al. The impact of air pollution on respiratory diseases in an era of climate change: A review of the current evidence. Science of The Total Environment. 2023;898:166340.

5. Samad A, Garuda S, Vogt U, Yang B. Air pollution prediction using machine learning techniques – An approach to replace existing monitoring stations with virtual monitoring stations. Atmospheric Environment. 2023;310:119987.

6. Zhang Z, Zhang S, Chen C, Yuan J.

A systematic survey of air quality prediction based on deep learning. Alexandria Engineering Journal. 2024;93:128-41.

7. Silibello C, Bolignano A, Sozzi R, Gariazzo C. Application of a chemical transport model and optimized data assimilation methods to improve air quality assessment. Air Quality Atmosphere & Health. 2014;7.

8. Pokharel S, Ghimire P. Data-driven MLmodels for accurate prediction of energy consump- tion in a low-energy house: A comparative study of XGBoost, Random Forest, Decision Tree, and Support Vector Machine. Journal of Innovations in Engineering Education. 2023;6:12-20.

9. Zhou S, Wang W, Zhu L, Qiao Q, Kang Y. Deep-learning architecture for $PM_{2.5}$ concentration prediction: A review. Environmental Science and Ecotechnology. 2024;21:100400.

10. Venkateswaran D, Cho Y. Efficient solar power generation forecasting for greenhouses: A hybrid deep learning approach. Alexandria Engineering Journal. 2024;91:222-36.

11. Kamali Mohammadzadeh A, Salah H, Jahanmahin R, Hussain AEA, Masoud S, Huang Y. Spatiotemporal integration of GCN and E-LSTM networks for $PM_{2.5}$ forecasting. Machine Learning with Applications. 2024;15:100521.

12. Gulia S, Khanna I, Shukla K, Khare M. Ambient air pollutant monitoring and analysis protocol for low and middle income countries: An element of comprehensive urban air quality management framework. Atmospheric Environment. 2020;222:117120.

13. Rahman A, Usama M, Tahir M, Uppal M. Data driven framework for analysis of air quality landscape for the city of Lahore. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2022;XLVIII-4/W5-2022:167-73.

14. Waqas M, Humphries U, Chueasa B, Wangwongchai A. Artificial Intelligence and Numerical Weather Prediction Models: A Technical Survey. Natural Hazards Research. 2024.

15. Sherstinsky A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena. 2020;404:132306.

16. Wang X, Huang T, Zhu K, Zhao X. LSTM-based broad learning system for remaining useful life prediction. Mathematics. 2022;10(12):2066.

17. Zhang J, Du J, Dai L, editors. A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. 2017 14th IAPR international conference on document analysis and recognition (ICDAR); 2017: IEEE.

18. Alkawaz AN, Kanesan J, Khairuddin ASM, Badruddin IA, Kamangar S, Hussien M, et al. Training Multilayer Neural Network Based on Optimal Control Theory for Limited Computational Resources. Mathematics. 2023;11(3):778.

19. Aslan S, Zennaro F, Furlan E, Critto A. Recurrent neural networks for water quality assessment in complex coastal lagoon environments: A case study on the Venice Lagoon. Environmental Modelling & Software. 2022;154:105403.

20. Tyagi A, Rekha G. Challenges of Applying Deep Learning in Real-World Applications. 2020. p. 92-118.

21. LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436-44.

22. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012;25.

23. Hinton GE, Osindero S, Teh Y-W. A fast learning algorithm for deep belief nets. Neural computation. 2006;18(7):1527-54.